Spring 5-20-2020

USING DEEP LEARNING AND LINGUISTIC ANALYSIS TO PREDICT FAKE NEWS WITHIN TEXT

John Nguyen

San Jose State University

USING DEEP LEARNING AND LINGUISTIC ANALYSIS TO PREDICT FAKE NEWS WITHIN TEXT

A Thesis

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

John Nguyen

March 2020

The Designated Thesis Committee Approves the Thesis Title

USING DEEP LEARNING AND LINGUISTIC ANALYSIS TO PREDICT FAKE NEWS WITHIN TEXT

By

John Nguyen

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2020

Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Vinh Phuong	Team Financial Advisor

ABSTRACT

Using Machine Learning and Linguistic Analysis to Predict Fake News within Text

By John Nguyen

The spread of information about current events is a way for everybody in the world to learn and understand what is happening in the world. In essence, the news is an important and powerful tool that could be used by various groups of people to spread awareness and facts for the good of mankind. However, as information becomes easily and readily available for public access, the rise of deceptive news becomes an increasing concern. The reason is due to the fact that it will cause people to be misled and thus could affect the livelihood of themselves or others. The term that is coined for spreading false information is known as fake news. It is of the utmost importance to mitigate this issue, thus the proposition is to perform a study on technological techniques that are being used to prevent the spread of dishonest and propagandized information. Since there are an abundance of websites and articles that internet users could read, the use of automated technology was the only logical option when dealing with fake news. The techniques that were used in this study were based around linguistic analysis and deep learning. The end objective was to create a classifier that was able to judge an article based on the amount of fake news within it. Experiments were performed on these classifiers, which tried to prove that applied linguistic analysis was important in improving the accuracy of the classifiers. The results from this study displayed evidence that applied linguistic analysis did not have a sufficient impact, whereas deep learning and dataset improvements did have an impact.

Keywords: Linguistic Analysis, Deep Learning, Fake news, Classifiers, Text

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Robert Chun for giving me the ability to pursue a project that was enjoyable and enriching in my pursuit of knowledge. I would like to thank my committee members Dr. Thomas Austin and Dr. Robert Chun for being amazing professors and teaching me new topics so that I may apply them to this project. I would like to acknowledge all of my professors for teaching me new ways to apply computer science in applications and programs. I would like to finally acknowledge all of my friends and family for supporting me through my Graduate Program.

TABLE OF CONTENTS

CHAPTER

CHAPTER 1

Introduction

The very simple idea of news was to obtain information about current events. There have been many methods to obtain information, which range from conversation, printing press, newspaper, radio, television, and internet. The key point to take away from all of these methods was the speed at which information was able to travel. For instance, during the use of the printing press, news was very slow due to the fact that it required an extraneous amount of time to gather information, print the news, and distribute it [1]. At that point in time, the news that was printed could take a few weeks to reach a wide audience [1]. This would mean that a small mistake in facts would be disastrous for the public and the news outlet because it would take an equal amount of time to fix the error and to reinform the public on the correct facts. The benefit of the printing press was that not all people had access to it, so news sources were generally reliable. The rise of the internet era brought about a tremendous decrease in information travel time. The news was able to reach a wide array of people from across the globe with little to no effort, but with this new mode of transportation, it brought about ease of access for every internet user to spread the news. This effect would cause an increase in news that are deemed untrustworthy and false. The term coined for this type of news was "fake news" which was based on the idea of deliberately spreading disinformation to people via social media, news outlets, or other quick means. The key goal of disinformation was to sway public opinion on certain topics. To understand the topic of "fake news", we need to first understand how news was generally

viewed throughout history and how rising technology brought about a stronger need to control the media.

1.1 The Definition of Fake News

The concept of fake news was not considered a new thing in the history of news because other terms that could be used for fake news is misinformation, hoaxes, propaganda, satire, and disinformation [2]. The main goal of fake news was to write and publish information that could mislead the people who read it. With this type of misdirection, it would cause damage to any interested party that was tied to the topic of fake news. For example, I could write up an article on anti-vaccination for my Facebook friends to view. The article that I am writing would be misleading since it would state that vaccination could cause malformed diseases. A few of my Facebook friends could then like and share the article amongst their friends, thus the idea of anti-vaccination was spread within a community. By undermining scientific data, fake news was able to make it difficult for serious coverage of any topic. There was considered to be around 7 types of scope that fake news can fall under, which are satire, misleading, imposter, fabricated, false connection, false context, and manipulated content [3]. The following table [Table 1] gives a broad definition of each type of form that fake news can take.

Satire/Parody	No intention to cause harm but has the potential
Misleading Content	Misleading use of information to frame issue or individual
Imposter Content	Real new sources are impersonated
Fabricated Content	New content is confirmed false and designed to deceive
False Connection	Visuals and titles do not support the content
False Context	Real news has false contextual information
Manipulated Content	Genuine information is manipulated to deceive

Table 1: The 7 Types of Fake News [3]

The sharing of information had always been a concern for the news since its inception, but with the advent of social media platforms such as Facebook and Twitter, it has been easier to inform a mass audience on any topic. The speed at which information travels through these platforms was much faster due to the tendency of sharing information that was new or surprising. The verification process of the posted information was done by the user, thus fake news had higher a chance of being on the platforms. By having more voices in the public, it has caused people to build distrust within the mainstream media because unlike media companies who have to make money, the people on social media platforms do not have that constraint. This divide in principle caused the public to be less trusting to mainstream media and more trusting towards independent sites and individuals. According to a Gallup poll in 2016, only 32% of people polled trust the media, whereas in 1974 there was around 70% [4].

1.2 Example of Fake News

Fake news was hard to detect because the sources used to create the information might be trustworthy, but the way that the news was worded and expressed could cause a different meaning. The following statement [Figure 1] was a verified fake statement from POLITIFACT.COM. We will analyze the statement on the key points that make it considered to be fake news.

> *"Go to your favorite grocery store and buy Cream of Tartar Seasoning and a gallon of Orange Juice. Mix 1 teaspoon in a glass and drink once a day. I recommend when you up and another glass halfway through your day. I know this sounds too simple, but it really works! The cream of tartar flushes the nicotine out of your system and blocks it from receiving it again! After about two days, smoking tastes like s***, you're blocked from the nicotine rush and the desire is gone!"*

Figure 1: Example of a Fake Facebook Statement from POLITIFACT.COM [5]

The purpose of the Facebook post was to inform people with the idea that drinking the concoction of cream of tartar and orange juice will help stop smoking addiction. The concept that the post relies on was the idea that detoxification diets will help remove toxins from the body. This statement was considered a half truth from the National Center for Complementary and Integrative Health office of the U.S. Department of Health and Human Services, but there has been no definitive study on the subject [5]. The intention of the post was to help people and not do harm, so this type of fake news will fall under satire. The reason this was considered satire was because drinking the mixture will not harm the reader, but it will make them look like a fool into believing that smoking addiction can be fixed by detoxification. The key points that can help

determine the post's fakeness was the use of exclamation points, lack of evidence/sources, and lack of defined instructions. The exclamation point at "but it really works!" give off an excited emotion to say that the mixture will guarantee to work. As a result, it tells the reader to trust the writer without any evidence to back up the claim. This would put the burden to find proof onto the reader. The other issue with the statement was that the instructions were not written in a list format, which shows lack of trust with the recipe. The main significance of the post was that it was shared over 300,00 times since October 14th, 2019 [6]. This would mean that each of those shared posts are unique users, therefore putting the amount of people that saw the post significantly over 300,000. The Facebook post was meant as a method to help people, but even good intentions can be fake news.

1.3 Example of True News

The next statement [Figure 2] will show a verified true statement that was posted on Twitter from West Virginia's senator Joe Manchint. We will analyze the key things that make the statement truthful and not fake news.

Figure 2: Example of Twitter Statement that is Truthful from POLITIFACT.COM [6]

The purpose of the Twitter statement [Figure 2] was to inform constituents that the Senator was trying to address the issue with increased child homelessness in West Virginia. The key points that make this statement truthful, when compared to the Facebook statement [Figure 1], was that the Senator started off by giving proof of which State Department contains the homeless youth statistic of over 10,000. The Senator then mentions that he wants to address the

issue by sending a letter to the U.S. Secretary of Education. He gives a link to the Secretary's official Facebook page so that readers know who he is referring to. He lastly included a link to the letter that he was sending as a proof of transparency. The link that he used was a government website, which was generally credible and reliable for sources of information [6]. The Twitter statement was a good example of how truthful news can be delivered on social media platforms, which was through a combination of reliable sources, statistics, and speakers. The negative aspect of the statement was that the sources used did not directly link to the statistical evidence written in the statement, but instead an official Facebook page of the department. This means that the readers would still have to find data on the 10,000 homeless youth statistics. The main difference between this Twitter statement and Facebook statement is that the Facebook statement did not have an author to verify the information. The Facebook statement utilized the shared mentality where friends of the shared post will oftentimes believe in the post because their friend was the one sharing it [Figure 1]. The opposite was true for the Twitter statement where the credibility of the statement was put on a single U.S Senator that relies on being truthful to the public [Figure 2]. Distributing real news was a difficult task, but when done right, it builds trust between the reader and writer. However, this does not mean that readers can blindly trust news sources, and will have to do their due diligence when checking for fake news.

CHAPTER 2

Background of the Technologies

2.1 Artificial intelligence, Machine learning, Deep learning

The concept of artificial intelligence(AI) comes from the idea of building a smart computer that was capable of doing tasks that usually require the assistance of humans. The main goal of AI was to solve the Turing test, which was a test designed by Alan Turing in 1950 [7]. The test was designed to determine whether or not a computer performing the test was intelligent. The test was similar to an imitation game where there are three isolated players to the game, one being a computer. One of the human players sits in an isolated position and is interrogated by a computer player and a human player. The goal of the player being interrogated was to determine which interrogator was the computer. If the interrogated player chooses wrong, then it was determined that the computer can think intelligently. Artificial intelligence has been around for a long time and there have been many unique methods that are used to create an AI program that can think for itself. The main two concepts to consider when discussing AI was machine learning and deep learning. These two concepts are not separate from AI, but instead subsets of one another where machine learning was a subset of AI and deep learning was a subset of machine learning [8].

Machine learning was an area of AI that focuses on the idea that a computer system can use data models to make decisions without human interference or being explicitly programmed [9]. To put it simply, a software application will utilize algorithms that are able to receive data and then use mathematical analysis to predict an output. The machine learning model was very

similar to data mining and predictive modeling where the main difference being that one was making a decision while the other was predicting future outcomes [10]. An example of a widely used machine learning application was the recommendation engine where machine learning algorithms use data of previous online purchases and searches to display personalized ads. Other examples of areas that machine learning could be used was spam detection, fact checking, security, and news feed. The ways that machine learning does the teaching of computer systems was through supervised and unsupervised learning [9]. The supervised learning utilizes a dataset where the input and output are labeled by the programmer that was designing the system. The next step of this learning method was that a mapping function was created so that when a new input was added, the system was able to make a decision to determine the properties of the input [9]. The result of that decision will be feedback to the main system in an iterative process until a certain performance is reached that matches the designer's requirements. An example that might be used for supervised learning was when a system was trained on datasets of cat and dog images. After the training phase, the system was able to identify unlabeled images of a cat, dog or neither. The unsupervised learning utilizes datasets that do not have labeled input data, but instead it reviews data through multiple layers and then reaches a conclusion. Common usage of unsupervised learning was clustering analysis, which was used to identify data patterns in very large datasets. The end goal of unsupervised learning was similar to supervised learning where we want to have a repository of associations for the purpose of analyzing new data that the users input [9]. The following table [Table 2] shows details of machine learning models that might be used to build classifiers.

Machine Learning Models	Details
Linear Regression	A classic model in finding the linear relationship between continuous values.
Logistic Regression	A statistical model to predict whether or not a given object belongs to a certain class. This model usually relies on a binary dependent variable such as pass or fail.
Decision tree	For this model it uses a tree-like structure of decision and consequences. The model is followed by an explicit representation of decision and decision making through a question that is either a yes or no answer.
K-Nearest Neighbors	In this method the data points are arranged in a space where a K-value is used to determine the similar data points of an input case.
Random Forest	The model consists of a large number of individual decision trees that work together as an ensemble where each decision tree will vote on a prediction based on what they output.
Naive Bayes	The model uses the bayes theorem to create a probabilistic approach that is based on the variables in the dataset. For instance, if the word "fake" is used 20 times in the article, then the article will be classified as such.
Neural Network	A neural network model is where a function can have multiple weighted inputs that is followed by an output. The output of the function is able to be the input for another function, which is what creates the basis of deep learning.

Table 2: Examples of Machine Learning Models [8, 9, 12]

Deep learning is the last topic to discuss and it will be the most important since the focus of the experiment will utilize a deep learning algorithm for fake news detection. The simple concept of deep learning focuses on utilizing numerous neural network layers of machine

learning algorithms to provide a structure of complexity and abstraction [11]. An example of how to interpret deep learning versus machine learning was to think of an object and point toward the object. A question that will be asked in a machine learning algorithm would be whether the object was a table or not. This type of thinking determines how machine learning does its training, but for deep learning the question extends to other aspects of the object until the system gets a decent idea of the object's properties. For instance, the system could point toward a wooden table which adds wood as one of the properties of a table, then a metal table would have metal. This concept of learning was similar to how the human brain processes data by creating new patterns of an object and then making a decision based on the learnt patterns [8]. The end result of deep learning is to constantly build layers of abstraction from previous levels of knowledge within the system [Figure 3]. Each level in the system utilizes an algorithm that processes a nonlinear transformation of the input it receives, then a model is given as an output for the next level of iterations until a satisfactory performance level was reached for the system [11].

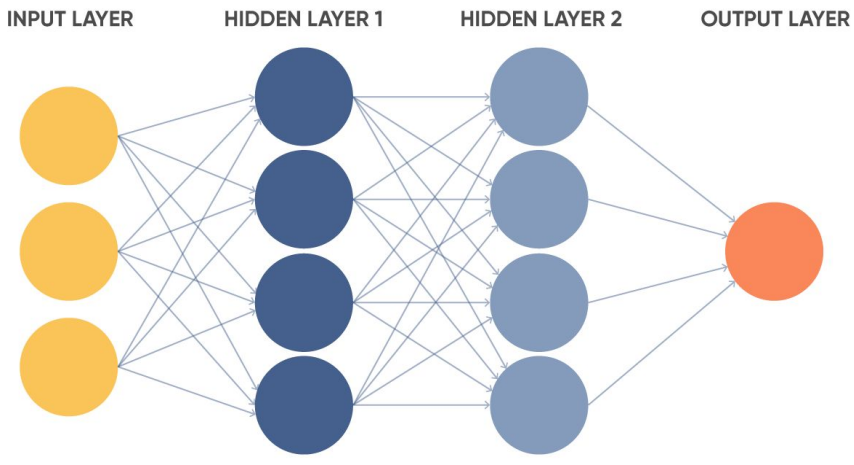

Figure 3: Diagram of the Hidden Layers in Deep Learning [8]

CHAPTER 3

Literature Survey of Fake News Experiments

3.1 Deep Learning Algorithm for Detecting Fake News in Text

The first experiment that we will explore is the research experiment "Deep learning Algorithms for Detecting News in Online Text" (Sherry Girgis et.al 2019) [13].The objective of this research article was to build a classifier that can predict fake news based solely on its content. The main approach to the problem was purely through deep learning avenues such as the RNN and LSTM models. The datasets that were used in the experiment were split into two categories where the positive sets were marked as truthful statements and the negative set was marked as false statements. The positive dataset was collected by the research group from a tested, valid method. A supplemental dataset was used in the experiment aptly named LIAR which was a readily available dataset (William Yang et.al 2017) [13] for fake news classifiers. The specifics of the LIAR data set was that it contained around 13,000 short statements from POLITIFACT.com, which was then manually labeled based on various identifiers [13]. Some examples of the identifiers used for the dataset were truthfulness, subject, context, speaker, and state. The following [Table 3] contain examples of the exact detail that would be in the dataset. A thing to note is that the "count" category was split into 5 sub-categories which are barely true, false, half-true, mostly true, and pants on fire. The reason for these categories was to determine the truthfulness of the statement.

Label: True	**Label:** false
Statement: Building a wall on the U.S. - Mexico border will take literally years.	**Statement:** Wisconsin is on pace to double the number of layoffs this year.
Subject: Immigration	**Subject:** job
Speaker: rick-perry	**Speaker:** katrina-shankland
Speaker's job title: Governor	**Speaker's job title:** state representative
State: texas	**State:** wisconsin
Party: republican	**Party:** democrat
Counts: 30 30 42 23 18	**Counts:** 2 1 0 0 0
Context: radio interview	**Context:** a news conference

Table 3: Examples of Data in the LIAR Dataset [14]

After the dataset collection, the data were pre-processed so that it would work with the system that the group built. The data cleaning used in the experiment was splitting, stopwords, and stemming. The splitting portion separated the statements into separate sentences, then stopwords were removed from each sentence, and lastly stemming returned each leftover word to its origin. A deception system was created by the group to give each word a vector that represents latent features of a word [13]. For instance, the word "King" might have a vector number of 0.99 for royalty and masculinity whereas the word Queen might have 0.99 for royalty and only 0.05 masculinity [15]. By adding these weights to a word, it will result in contributing to the definition of the word for deep learning models. The flow of the experiment [Table 4] goes as follows:

First step	Data Preparation (splitting, remove stopwords, stemming)
Second step	Stemmed words is the input for word vector system
Third step	Results of the second step are the input of the Vanilla, GRU, and LSTM
Fourth step	Determining truthfulness of a small piece of news

Table 4: Experiment Steps for "Deep Learning Algorithm for Detecting Fake News in Text" [13]

Model	Test Accuracy
CNN	0.270
Vanilla	0.215
GRU	0.217
LSTM	0.2166

Table 5: Results of the Experiment for Vanilla, GRU, LSTM, and CNN [13]

The result of the table [Table 5] shows a comparison between Vanilla, GRU, and LSTM experiments. It was stated that GRU had the best result due to being easier to train, and better performance. The CNN data point was noted as a comparison with another experiment (Wiliam Yang et.al, 2017) [13]. The key takeaway from this article was that other models for deep learning are less optimal than CNN, therefore it is best to focus on the CNN model for a fake news classifier. Another noteworthy thing about the article is the dataset and data preprocessing, because in order to make a fake news classifier, a good set of training data is needed.

3.2 Fake News Pattern Recognition Using Linguistic Analysis

The experiment for this article took a different approach when compared to the previous articles. This article focuses more on the linguistic analysis aspect rather than the design of an all encompassing fake news classifier [16]. The main algorithm used in the experiment was the K-nearest neighbor, which classified fake news and compared them to credible news sources. The dataset that was constructed for the experiment utilized true and fake articles on "Hilary Clinton" that media company Snopes labeled as false [16]. The main points that the experiment looked in an article was for ambiguity, abusiveness, subjectivity, and deceptiness. Part of analyzing the data utilized sentiment analysis versus length of the article to establish associations between features of fake and true articles. The first step in the data preprocessing was to find important words, which used the NLTK framework for the natural language process to assign parts-of-speech to each word token. The next step in the data preprocessing for the article was to prevent the splitting of tokens so that inherent meanings are intact. The data for the sentiment analysis of the datasets shows that credible articles had 18% negative sentiment, 44% positive sentiment, and 38% neutral [16]. For the fake articles there was 56% negative sentiment, 17% sentiment, and 27% neutral [16]. These two statistics were used as markers for the experiments to form a predictive model. The next model made for the algorithm was a bag-of-words where words frequencies of noun phrases are filtered. The results found that true statements are specific, objective, and critical of their topic. The result on fake news shows that the statements used ambiguity, abusive, and polarized language. The sentiment analysis helps reveal the hidden bias that was reflected in an article. After all of the data processing and analysis was done on the

dataset, the K-nearest neighbor algorithm was applied. The results of the algorithm showed it was able to predict a fake statement, but with only an accuracy of 66.66% [16].

The experiment reveals that the task needed to preprocess linguistic data is extensive and vast. There are many unique models that linguistic analysis can use to clean data before an algorithm is used. The downfall of the experiment was the amount of data used for K-nearest neighbor to work properly. Thousands of data points are needed to properly plot and test the algorithm. The takeaway from this experiment was to use a variety of linguistic analysis before inputting the resulting data into a machine learning system.

CHAPTER 4

Hypothesis and Dataset

4.1 Hypothesis

Linguistic analysis could be used to build and improve the learning accuracy of a neural network learning model. The linguistic analysis will take statements from a fake news dataset and perform certain modifications on the data so certain elements of the text statements are extracted. The learning model will train with these modifications and result in an accuracy percentage for that version of the training set. The result of the linguistic analysis testing will take the best increase in percentage gain and combine the modification into one learning model that will be used to compare against commercial products, which will show that linguistic analysis is important when dealing with natural language processing for fake news.

The linguistic analysis that will be used in this project will be word embedding, stopword removal, stemming, lemmatizing, part of speech tagging, sentiment analysis, and multidata categorizing. The learning that will be needed in this project will be based around a simple CNN, LSTM, and CNN + LSTM. These modifications and learning models will be explained further in the chapter that details the experiment.

4.2 Dataset

The dataset that was used for this project was based around the LIAR dataset, which consists of around 13,000 tuples. This first dataset was cleaned of issues that resulted from the conversion of a tsv(tab separated value) file to a csv(comma separated value) file. The columns

in the original tsv file were reduced to only the statement and label for the csv file. The table [Table 6]below shows the first five tuples in the dataset.

news	sentiment
Says the Annies List political group supports third-trimester abortions on demand.	false
When did the decline of coal start? It started when natural gas took off that started to begin in (President George W.) Bushs administration.	half-true
Hillary Clinton agrees with John McCain "by voting to give George Bush the benefit of the doubt on Iran."	mostly-true
Health care reform legislation is likely to mandate free sex change surgeries.	false
The economic turnaround started at the end of my term.	half-true

Table 6: Initial 5 tuples of the LIAR dataset

A second dataset was created with the same data as was in the first dataset. The reason this second dataset was created was due to the inclusion of sentiment analysis and thus more data was added to complete this section of the project. Four additional columns were added to the dataset, which added weights to the statement based on its sentiment value of either being negative, neutral, or positive. The fourth column was for compounded values that portrays the sum of all lexicon values of the negative, neutral, and positive values. For instance, if a compound value was at 0, then the statement is neutral sentiments while a negative compounded value meant that it was negative sentiment and vice versa for positive values. The following data

table [Table 7] shows the additional columns that were added to the second dataset on the first five tuples of data.

negative	neutral	positive	compound
0.115	0.692	0.192	0.25
0	0.902	0.098	0.3612
0.107	0.687	0.206	0.3182
0	0.606	0.394	0.7579
0	1	0	0

Table 7: Initial 5 tuples of the LIAR dataset for sentiment values

A third and fourth dataset was made for the project which was a different subset of the LIAR dataset. This meant that the data for this part was completely different from the first two datasets used in this project. The reason for this dataset inclusion was to demonstrate that the accuracy data obtained in the experiments were not tied to the data. So for instance, if the accuracy for one of the tests in this dataset was similar to the data obtained in the first dataset, then it can be implied that data variance was not a factor in determining the accuracy percentage. If there was a significant difference between the two datasets, then it will conclude that data differences had a major impact on the results. Further discussion about this part of the

experiment was included in the later chapters. The limiting factor in utilizing this dataset was that the amount of tuples were significantly less than the first dataset. There were a total of around 1,200 tuples created for this dataset. The same procedure was applied when modifying this dataset as the previously discussed dataset.

The last dataset made for the project utilized a Kaggle dataset[17] for fake news named "Getting Real about Fake News." The dataset had only fake news text, therefore the Kaggle dataset needed to be combined with only the true news of the LIAR dataset to form a mixed dataset. The labels for the Kaggle dataset[17] were all set to false, since that dataset did not have significant label distinction. Due to this lack of distinction, the "pants-fire" label was not included in the mixed dataset. The lack of 6 labels for the dataset affected parts of the modeling code for certain experiments, but the changes did not affect the learning model in a significant way.

CHAPTER 5

Details of the Experiments

5.1 Keras and Word Embedding

The first part of the project was developing the neural networks in python. The main framework used for this part of the project was Keras which was described as an open sourced neural network library that allows for fast development of deep neural networks for experimentation and prototyping [18]. The reason why Keras was chosen over other deep learning platforms was due to its four guiding principles of user-friendliness, modularity, extensibility, and python integration [18]. The user-friendliness was displayed in the Keras API design where it was easy to start development based on common design patterns of deep neural networks. The modularity of Keras was in the creation of the models with the use of layers, optimizers, and activation functions. These API calls allow for the creation of complex models through the use of threading different or similar independent modules. The extensibility of Keras allows for custom creation and modification of modules such as new recursion layers for model creation [18]. This aspect of Keras was considered a useful aspect of the framework, but was not fully utilized in this project. The benefit of python integration in this framework was that it allows for quick and simple access to unique and different libraries for linguistic analysis, such as sentiment analysis and NLTK. By utilizing most features of Keras, the development of the deep learning models was straightforward.

An important tool that was used for all of the following learning models was word embedding. The concept of word embedding was to have similar representation of similar

meaning words. The end result of this embedding would be an understanding of the context, semantic, and relation of a word. One of the main basic ideas was to have each word mapped to a vector of numerical values, which is why another name for word embedding was word vector. The vector from this representation would need to be a dense representation with hundreds of dimensions. An example would be with the words “bad” and “awful” where embedding their vectors will have similar spatial positions such as [0,1,0,0,0] and [0,0,1,0,0]. If word embedding did not exist, then the word “bad” and “awful” would have different vectors, which we know would be incorrect since we know the meaning while a model without prior knowledge will not [19]. The main source for all of this project word embedding comes from GloVe, which stands for Global Vectors for Word Representation. The algorithm of GloVe is a subdivision of word2vec, another type of word embedding tool, that focuses on defining the context on the entire text corpus rather than in a local area [20]. The conclusion of this type of learning model would be a better word vector, which is why it was chosen as the embedded layer of this project.

5.2 Convolutional Neural Network

Since the Keras library had heavy support for convolutional and recurrent neural networks, these two neural networks were chosen as the learning models. The convolutional neural network (CNN) was the first deep learning model created. A convolutional neural network utilizes convolutional layers as building blocks for its model. The ideal use for convolution neural networks was for the classification of images because of its features to differentiate objects of an image through the use of several layers [21]. However, an image still needs to be converted into a matrix of pixel values since a computer was not able to naturally look at an

image and see a physical object. For natural language processing, the use of word embedding was able to convert a statement into a similar matrix of values, thus giving the full capabilities of CNN to natural languages. The principal design of convolutional neural networks was that it employed regularized hidden layers between the input and output layers. The main concept behind hidden layers was that each node in one layer was fully connected to nodes in the next layer which was commonly known as multilayer perceptrons [22]. The downside to multilayer perceptrons was that data tends to be overfitted, therefore making prediction unreliable on unseen data. The unique aspect of CNN that separates it from other learning models was how it approaches regularization through the use of small, simple, and sparsely connected patterns [23]. This method of regularization allows for the assembly of complex modeling without each layer being fully connected therefore increasing the efficiency of training and resource management.

The first hidden layer within the CNN learning model was known as the convolution layer, whose job was to extract features from the input matrix. In this layer the matrix was convolved to form a smaller filter matrix known as a feature map [21]. This feature map allows the learning model to understand specific patterns and features of the matrix. The differences in data size can affect how a matrix was built, therefore padding was involved to pad the matrix with zeros to fit with the user defined matrix dimension. An additional part of the convolutional layer was the inclusion of non-linearity which meant that the values in the matrix needed to be non-negative linear values [21]. The nonlinear functions that are commonly used for deep neural networks were softmax, ReLU, tanh, and sigmoid. The main differences between these built-in functions were usually the performances and computation needs. The second hidden layer was known as the pooling layer, which had the purpose of reducing the number of parameters from

the feature map to a smaller matrix [21]. In essence, the goal of this layer was to keep the important identity and information of the matrix, while at the same time, reducing the matrix size. The common pooling layer that was done for CNN was max pooling which takes the largest element value in each matrix quadrant and input it into the new matrix. Other pooling includes averaging the value or taking the sum of all values. These two layers compose most of the hidden layers because the convolve and pooling were done for multiple rounds and sections of the input data until it reached the classification section of CNN. The classification part consisted of flattening out the many feature maps from the last section and forming a fully connected neural network similar to multilayer perceptrons [22]. An activation function, such as sigmoid, was the last step in allowing the model to classify objects.

The following snippet of code [Figure 4] shows how the CNN model was created with the Keras API where it employed the sequential Keras API to simply add layers to the model. The fit() method was responsible for getting the training data into the model for training. The X data represented the text statements that were converted to a tokenized form, put into array, and padded. The Y data represented the labels for each text statement in a 0 or 1 format such that 0 correlates to false and 1 to true. Lastly, the epoch defines the amount of passes over the training dataset, while validation split is the amount dedicated for the final score training to get the accuracy. The next two neural networks discussed will utilize the same structure as this CNN python code but instead with changes that match their model's architecture.

```
model = Sequential()
embed_layer = Embedding(vocab_size, max_length_embed_matrix,
weights=[embed_matrix], input_length=maxlen, trainable=False)
model.add(embed_layer)
model.add(Dropout(0.2))
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['acc'])
print(model.summary())
history = model.fit(X_train, y_train,
batch_size=var_batch_size_num, epochs=var_epoch_num, verbose=1,
validation_split=var_val_split_num )
score = model.evaluate(X_test, y_test, verbose=1)
```

Figure 4: Python code for Convolutional Neural Network

5.3 Long Short-Term Memory

The second deep learning model created for the experiments was long short-term memory (LSTM), which was a different version of recurrent neural network (RNN). The concept behind recurrent neural networks was that it was a feedforward network similar to CNN, but it had

internal memory to process a sequence of inputs [24]. The internal memory copies and stores the output from the previous run and uses the output on the next input. In other words, all inputs for an RNN model will always be connected with one another when compared to CNN independent inputs. This type of design makes it perfect for natural language processing because words might have different meaning and context when in a text statement. The downside in using an RNN model was that training and processing long sequences can prove to be a difficult task. The main problem with RNN models was the vanishing gradient problem where the problem states that gradients of the loss function will approach toward zero, thus making training difficult and inaccurate [24]. This problem will only affect certain activation functions that cause small derivatives, but more efficient functions such as ReLU do not have this problem. LSTM was a different version of RNN that mitigated the vanishing gradient problem without the need to change activation function [25]. LSTM had a cell state and three gates as part of its core building block to train a model. The gates are named as input gate, forget gate, and output gate. The purpose of the cell state was to act as a memory location for important information to be stored during training. The information that was stored within the cell state may be removed or added during any iteration of the training [25]. The three gates were the deciding factor in determining the relevancy of the data. The first gate was known as the forgot gate which decides on what data to add or remove by applying the sigmoid function on the hidden state and current value. If the output value of the sigmoid function was closer to 1, then the information is added to the training; otherwise, the information is forgotten [25]. The next gate was the input gate which determines the value that would be used to modify the cell state. The way this gate does this part was by getting the hidden state and current values and putting them in a sigmoid and tanh

function. The two values that were resulted from these two functions will be multiplied together where the sigmoid value was used in a way that picks the important value from the tanh function [25]. The cell state was the next block within the LSTM model, where the current cell state value was multiplied by the forget gate value and then added to the input gate value. The result of this block will be output of the new cell state. The last gate within LSTM was the output gate that decides on the hidden value for the next iteration. The previous hidden value and current value were put in a sigmoid function, while the cell state value was put within a tanh function. The result of these two functions were multiplied to get the new hidden state value for the next iteration [25].

5.4 CNN + LSTM

The last deep learning model used for the experiments was the combination of CNN and LSTM. This model takes the feature extraction from CNN and sequencing of LSTM to form a prediction model. The original name of this learning model was long-term recurrent convolutional network (LRCN) and its original intent was to handle spatial inputs like images [26]. However with word embedding, natural languages could easily be converted to form a spatial object that would then be able to fully utilize CNN + LSTM. The architecture design for this learning model had the initial input be used in the CNN model, then the output was put in the LSTM model [26]. The dense layer will still be similar to the other two models where it was a fully connected neural network for creating the classification portion of the model. In the end, the goal with this model was to create a different type of learning model that takes into account the features and sequencing of a statement.

5.5 Linguistic analysis: Stopword Removal

The first linguistic analysis done to the data set was removing stopwords from the statements and then applying the learning models to the updated data set. The concept of a stopword was a word that provides little to no context for a sentence; therefore, it was rational to remove those words from a sentence [27]. There were no major negative consequences when this type of linguistic analysis was done on a text statement. The slight downside to removing stopwords was that some words might actually be an important part of the sentence; however, that scenario will only affect a specific set of cases. An added benefit of removing stopwords was there will be an increase in performance during training because of the reduced size of the dataset. The following piece of python code [Figure 5] shows how the stoword was removed in the dataset. All of the dataset text statements were added to an array, split into individual words, and then compared to a list of english stopwords. The list of english stopwords comes from the python NLTK.Corpus library. If the checked word was in the stopword list, then the word was not added back into the statement. The analyzed statement was converted back to a string and pushed into the larger statement array for deep neural network modeling. The reason that stopword was an important test was because it was trying to show that less words in a dataset could maintain its meaning and form a more accurate learning model than the base model.

```
for k in J:
     stop_words = set(stopwords.words('english'))
     tokens = k.split()
     tokens = [w for w in tokens if not w in stop_words]
     X.append(list_to_string(tokens))
```

Figure 5: Python Code for stopword removal

5.6 Linguistic analysis: Stemming

The next two linguistic analyses were considered to be similar for natural language processing, but they differ in the sentences that they output. These two types of tools were part of the text normalization category and were usually known as stemming and lemmatization. The goal of stemming and lemmatization were to derive and reduce words to a common base form so that there was little variation between data [28]. The idea to add this concept to the experiment was to reduce over-variation so that the learning model was able to identify proper patterns and generalize for classification. Since stemming and lemmatization was considered different, the project split the two processes into separate entities and applied them independently on the same dataset. Stemming was the first process developed, and it was defined as a method of producing variants of a word to its root. For instance, the word "likes", "liked", and "likely" would all be stemmed to the word "like" [28]. For longer complex words, the stemming algorithm might cut the head and tail of a word so that it forms a word that resembles a root word. The words that might come out of this algorithm have a chance of not being a real word in the dictionary. An

example of this scenario would be "studies" being stemmed as "studi" because of the common occurrence of "es" in plural words. The errors that might result from stemming were known as overstemming and understemming. For overstemming, words are stemmed to the same root word even if they come from vastly different original words. For example, words like "university", "universal", and "universe" might all be stemmed to "universe," which would not be a good scenario because all of the words do not have similar meaning [29]. Understemming was the scenario with the opposite effect of overstemming where the words that were stemmed would be the same root word but instead results in two different root words. An instance where this scenario might occur was with the words "alumnus" stemming to "alumnu" , "alumni" stemming to "alumni", and "alumnae" stemming to "alumna" [29]. The reason these words will be stemmed these ways was due to the way that the english language spelled and defined these words. Another reason was due the fact that stemming algorithms operate on single words and not the entire context of the text statement. Therefore, to not discriminate against certain words, the stemming algorithm will ignore these special cases unless specified. The algorithm that will be used for the project was the porter stemmer algorithm, which was considered a fast and efficient algorithm with the downside being less precise than other algorithms. The porter stemmer algorithm function was called from the NLTK.Stem python library. The following code [Figure 6] shows that the statements were split into individual words and then stemmed by putting the words into the stem() function. The result of the stemmed words were concatenated back together and put into the larger array for modeling.

```
ps = PorterStemmer()
for l in J:
     tokens = l.split()
     sent = ""
     for y in tokens:
          sent = sent + ps.stem(y) + " "
     X.append(sent)
```

Figure 6: Python code for stemming words

5.7 Linguistic analysis: Lemmatization

Lemmatization was the other text normalization strategy that was similar to stemming where it brings together different forms of a word and puts it under one common word. The main difference of lemmatization, when compared to stemming, was that it takes the context of the word into account when determining the final result. Another important factor for lemmatization was that it considers all inflectional forms of a word whereas stemming does not. An example of this scenario was when the word "better" was lemma to "good" and "corpora" was lemma to "corpus" and not "corpo" [30]. The downside of lemmatization was that a dictionary, word meaning, and sentence context was necessary to properly output a lemma word [31]. Without these factors the lemmatization process would be worse than the steaming process. Due to the difficulty of lemmatization, the project only includes two forms of the strategy. The first form was just the standard lemmatization algorithm with the part of speech being set to a default value

of "verb." The following diagram shows how this process was done in python, which was by using the NLTK.Corpus library for wordnet and wordnetlemmatizer. The reason to use wordnet was due to the fact that it was a large and public lexical database for the english language and it offers lemmatization processes [30]. The second form included the proper identification of the part of speech so that the lemmatization process was to be more accurate. The diagram [Figure 7] below also reveals how the part of speech identification was done in python, which was through the NLTK library method of pos_tag() that takes tokenized statements as parameters and adds a part of speech to each word in the sentence. The tokenize statement was split into individual words so that they would be lemmatized with the correct part of speech. The overall goal and choice of lemmatization and stemming was to generalize statements so that the learning model did not have to worry about dealing with extreme outliers in the dataset. The importance of removing outliers was to prevent skews and misleading data representations.

```
lemmatizer = WordNetLemmatizer()
for l in J:
     postoken = word_tokenize(l)
     postoken2 = nltk.pos_tag(postoken)
     tokens = l.split()
     sent = ""
     i = 0
     for y in tokens:
          temp1 = postoken2[i][1]

          i = i + 1

          postemp = ""

          if(temp1.startswith('V')):

               postemp = wordnet.VERB

          elif(temp1.startswith('J')):

               postemp = wordnet.ADJ

          elif(temp1.startswith('N')):

               postemp = wordnet.NOUN

          elif(temp1.startswith('R')):

               postemp = wordnet.ADV

          else:

               postemp = wordnet.NOUN

          sent = sent + lemmatizer.lemmatize(y, pos=postemp) + " "

     X.append(sent)
```

Figure 7: Python code for lemmatization

5.8 Linguistic analysis: Sentiment Analysis

The last type of linguistic analysis done on the dataset was sentiment analysis which was ideal to identify and extract subjective information from a text statement. The interpretation of these text statements would be under the category of positive, negative, or neutral. The intent of sentiment analysis was to provide users and businesses the ability to determine customer's emotion toward a product based on customer interaction and feedback. The logic behind this concept was that there was too much unstructured data that analyzing them would prove to be difficult, thus sentiment analysis was used as a medium to process the different forms of data [32]. The most basic type of sentiment analysis was the rating process, which simply asks users how they feel about a product based on 5 stars being very positive and 1 star being very negative. This type of method was an easy solution of determining how users felt about a product, but this method was not always a good gauge. The reason was due to the fact that people's ratings were subjective and can be influenced based on other criterias and experiences. The issue would be similar for statements in natural language processing because the like to dislike ratio on a statement would be determined by the source that the statement originated from. To prevent the biases in rating systems, the experiment for sentiment analysis would not use that data as a factor of the learning models. The different types of methods that sentiment analysis algorithms could belong to are either rule-based, automatic, or hybrid [32]. For rule-based algorithms, sentiment analysis was accomplished through manually defined rules to identify the polarity and subjectivity of the statement. The algorithm may employ other linguistic analysis such as stemming, tokenization, and part of speech. However, if more rules are placed within the algorithm then the entire system could get complex and hard to maintain. The automatic

approach utilizes machine learning tools and processes to learn from sentiment data and create a classifier. The hybrid approach combines both the rule based and automatic method types to form a more accurate algorithm, but the downside was that the development for this algorithm was harder and complex. For the project, the algorithm utilized for sentiment analysis was the Vader Sentiment Analysis python packages, which was a rule based sentiment analysis tool that was designed to display sentiment data based around social media statements. The analysis would result in a score for positive, negative, neutral, and compound. The scores from the analysis were added as columns so that the deep neural network model could use them as input data. The difference with this part of the experiment was that the models employed multidata features to understand multiple columns of data. The following section will further describe how the multidata learning models functions in this project. Overall, the logic behind adding experiments and tests for sentiment analysis was due to the fact that english was a subjective language and that two logically similar statements might not have the same meaning due the way that they were spoken and interpreted. The score provided from a non discriminatory source would be able to provide more data so that the learning models does not have to deal with overfitting. The overfitting scenario will be when the model transitions to unseen data and cannot predict with a high degree of accuracy.

5.9 Multidata

In the experiment there were two types of multiple data methods that were used: the Y data that had multiple columns or Y data that were in one column but not in a binary format. For instance, the dataset used for the experiments had 6 types of labels for statements which meant

that the statement could have fallen into different categories. To make the experiment simpler a binary method was picked as discussed earlier. In these tests, each label had its own dense layer so that they could be more accurately identified and classified. The end goal with this section was to try and improve the accuracy percentage of the linguistic analysis tests.

The sentiment analysis and multidata tests were the only models that included the integration of multi-labeled data. For sentiment analysis models it used a method to concatenate two single arrays into a 2D array. The single array followed the same process that was used to convert int to binary format. The purpose of this was to use the compounded sentiment data as another criteria to classify the text statements. For the multidata models it had a Y data with either 5 or 6 labels depending on the dataset. The label encoder method was used to make an array of integer values based on the amount of labels in the column. For example, all instances of “true”, “false”, or “barely-true” would be set to 0 for “barely-true”, 1 for “false”, and 2 for “true.” The next step of label encoding would be categorizing the integers into a unique array of 0s and 1s. The purpose of this step was to allow the Y data to be passed into the fit method for modeling. The dense layer amount was also changed to reference the amount of new labels. The logic behind this change was so that the learning model could create different classifications for multiple labels. The end result would be a model that should be able to identify 6 different labels.

Chapter 6

Experiments and Results

6.1 Experiment: 1

The first experiment [Appendix I] accomplished was for a CNN model with a 100 dimension pre-trained word vector for Glove named Wikipedia 2014 + Gigaword 5. As the name suggests the pre-trained word vector used Wikipedia and Gigaword as a baseline to form the word vector of around 400K vocabulary and 6 billion tokens [33]. The tests done on this neural network consists of no linguistic analysis, stopword removal, stemming, lemmatize, and lemmatize combined with part of speech. The following diagram[Figure 8] shows the result of this experiment, which was the accuracy from training the CNN model on the Liar dataset. The accuracy was averaged from a series of 5 runs with 6 epochs. The average result was then multiplied by 100 to show a percentage score. By applying different linguistic analysis on the same CNN model, the data shows that there was not a significant accuracy change between the tests.

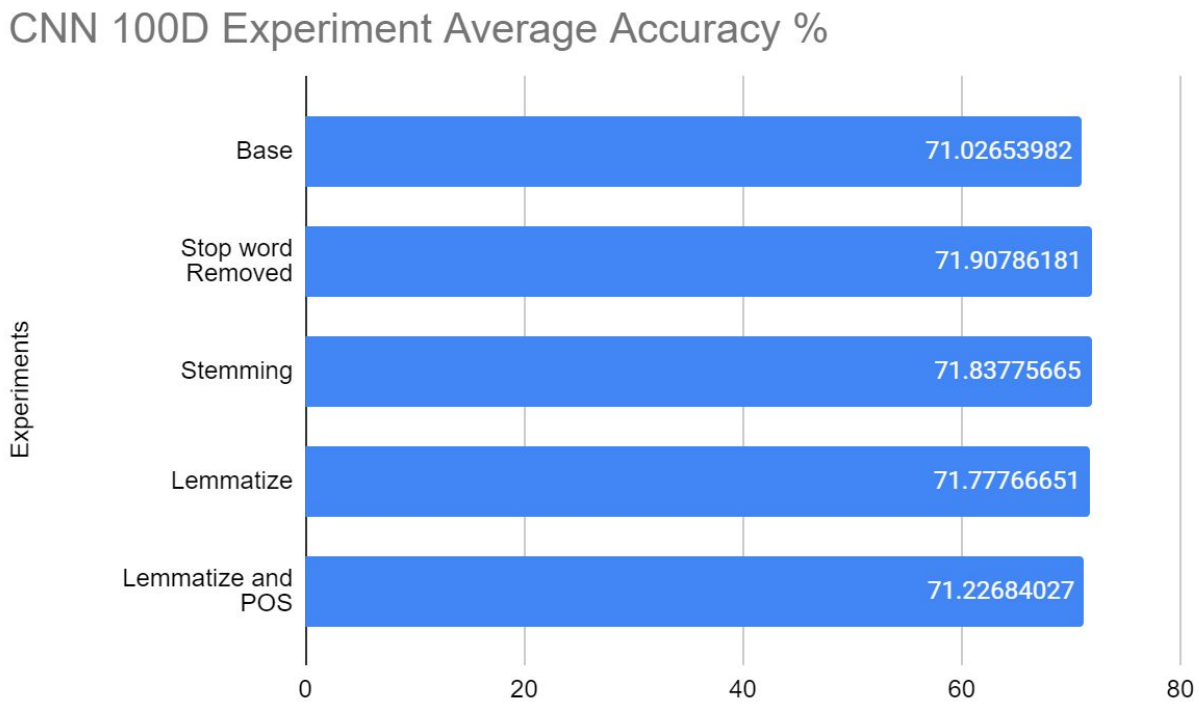

Figure 8: Experiment 1 Data

6.2 Experiment: 2

The next experiment [Appendix II] was accomplished on a different deep neural network, which was a LSTM model with the same 100 dimension pre-trained word vector. The experiment applied the same test as the CNN model, which was no linguistic analysis, stopword removal, stemming, lemmatize, and lemmatize combined with part of speech. The diagram [Figure 9] below shows the average result of this experiment, where the accuracy percentage obtained was still the same throughout all of the linguistic analysis. When compared to the CNN model the percentage was around 1% higher than the average score of the CNN with linguistic analysis. The purpose of this experiment was to show whether or not the CNN model or LSTM

model was better on the fake news dataset. The results reveal that the accuracy was only marginally better.

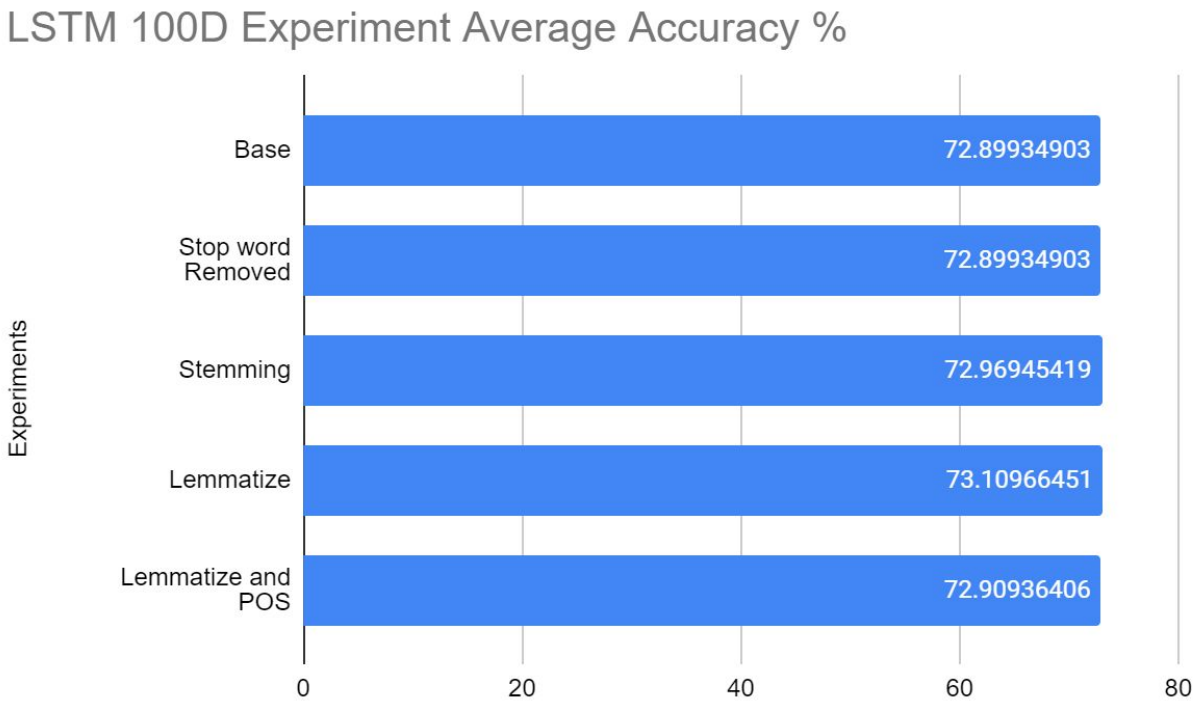

Figure 9: Experiment 2 Data

6.3 Experiment: 3

The third experiment [Appendix III] was on a CNN model combined with a LSTM model with the same 100 dimension pre-trained word vector. The same procedure and testing was applied as the CNN and LSTM experiments. The following diagram [Figure 10] shows the average results where some of the linguistic analysis accuracies were lower than the base model of no linguistic analysis. The notable ones were stopword removal and stemming. These changes

could mean that applying these linguistic analysis could hinder the model, but when looking at the CNN and LSTM diagrams, the data reveals that there was no reduction in accuracy for these two tests. This will then imply that the modeling for CNN + LSTM was not as efficient as CNN or LSTM separately. The purpose of this experiment was to apply a unique model that makes use of both CNN and LSTM, but with these accuracies, the modeling for the dataset was a little bit inconsistent when compared to the previous two experiments.

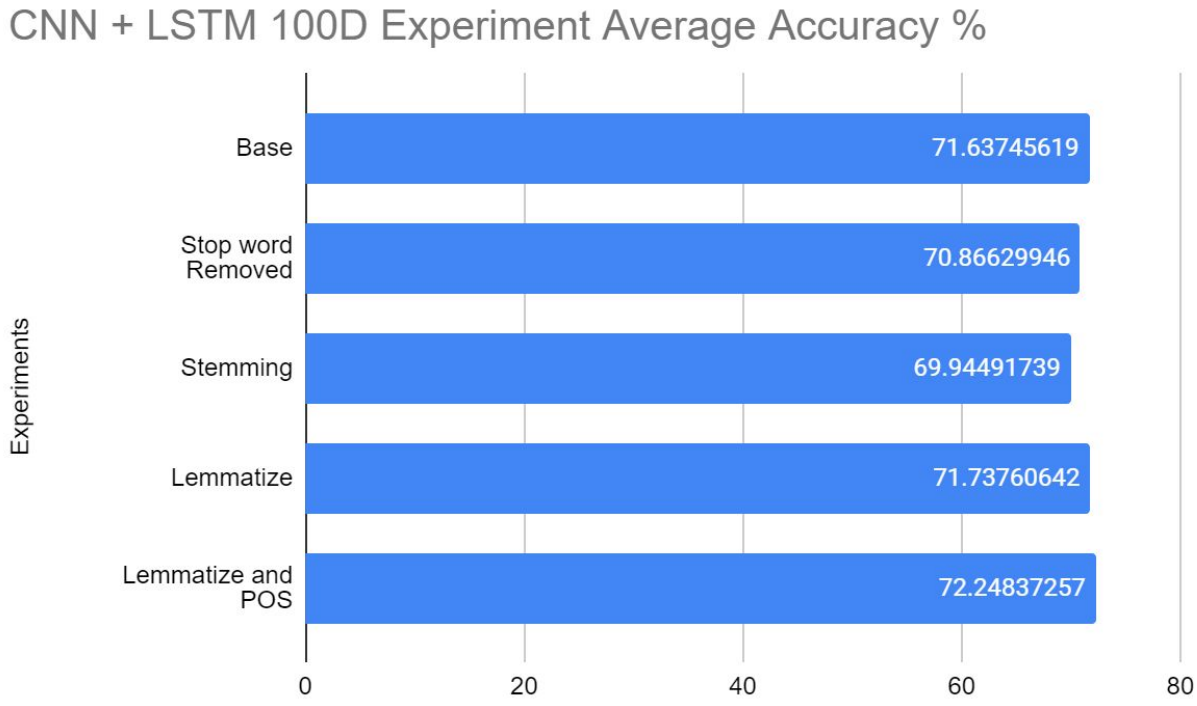

Figure 10: Experiment 3 Data

6.4 Experiment: 4

The fourth experiment utilized the LSTM model with a smaller dataset. The embedded layer utilized for this experiment was still the same as the previous three experiments. The smaller dataset was derived from the same LIAR dataset, but with around 1300 tuples. The results from the data table [Appendix IV] shows that the LSTM model was underfitting since the standard deviation was set to 0, thus making the result unsuitable for comparison. The reason for this experiment was to see if the size of the dataset matters in the LSTM learning model. The data confirms this suspicion and that a larger dataset was necessary for proper verification.

6.5 Experiment: 5, 6, 7

The next three experiments [Appendix V, VI, VII] utilized different pre-trained word vectors for the embedded layers, but with the same learning model of CNN. The 200 dimension Twitter word embedding was a pre-trained word vector that utilized 2 billion tweets, 27 billion tokens, and 1.2 million vocabulary [33]. The main reason to use this word vector was that it had 100 dimension and 200 dimension, which makes this word vector a good comparison for the Wikipedia 2014 + Gigaword 5. The following diagram shows the average accuracy percentage of the stopword removal test and the lemmatize with part of speech test. The data shows that the result was around the 70% range for all of the tests, which means that by applying different word vectors, the accuracy was not significantly impacted.

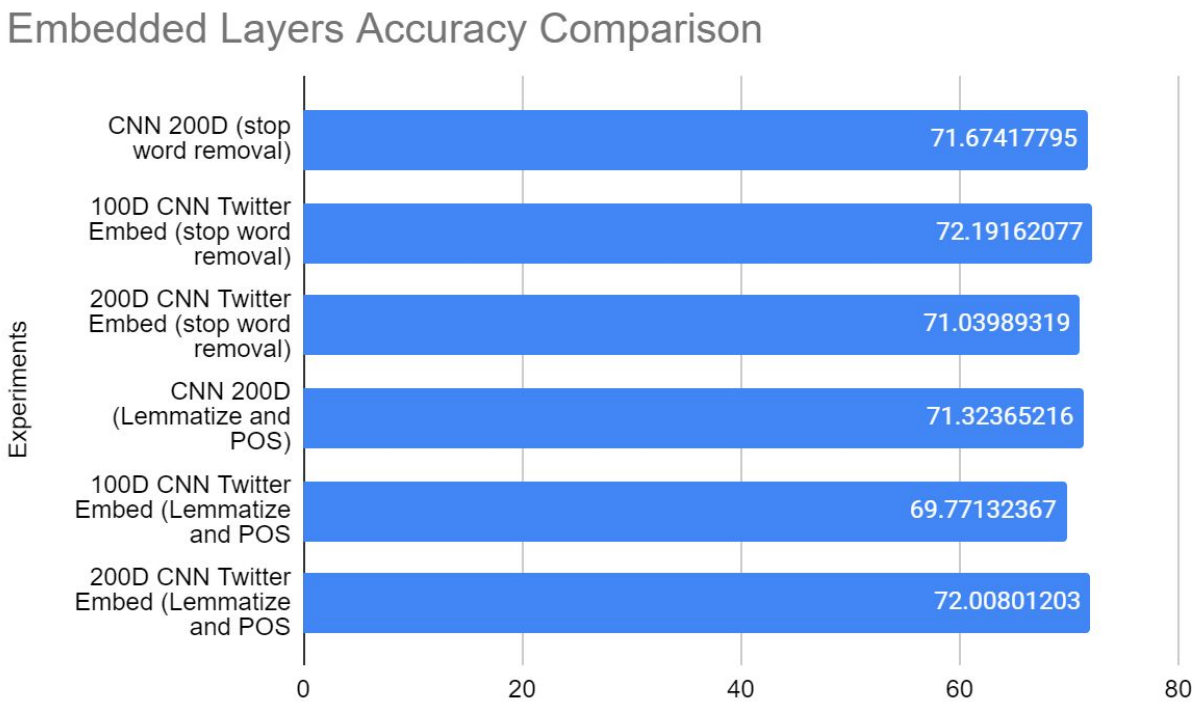

Figure 11: Experiment 5, 6, 7 Data

6.6 Experiment: 8

The last finished experiment [Appendix VIII] was done on a dataset of around 21,000 tuples. The dataset for this experiment used the modified Kaggle dataset [17]. The purpose of this experiment was to see if a larger size dataset would impact the accuracy of the same learning model as experiment 1. The following diagram [Figure 12] shows the result of the experiment based on the same parameters that were set for the first experiment. The data reveals that there was a significant change in accuracy percentage of around a positive 26% when the dataset was changed. However, the results also show that the linguistic analysis had little to no impact on

changing the accuracy percentage. The reason was due to the relative comparison of each linguistic test where there was only a difference of 1%.

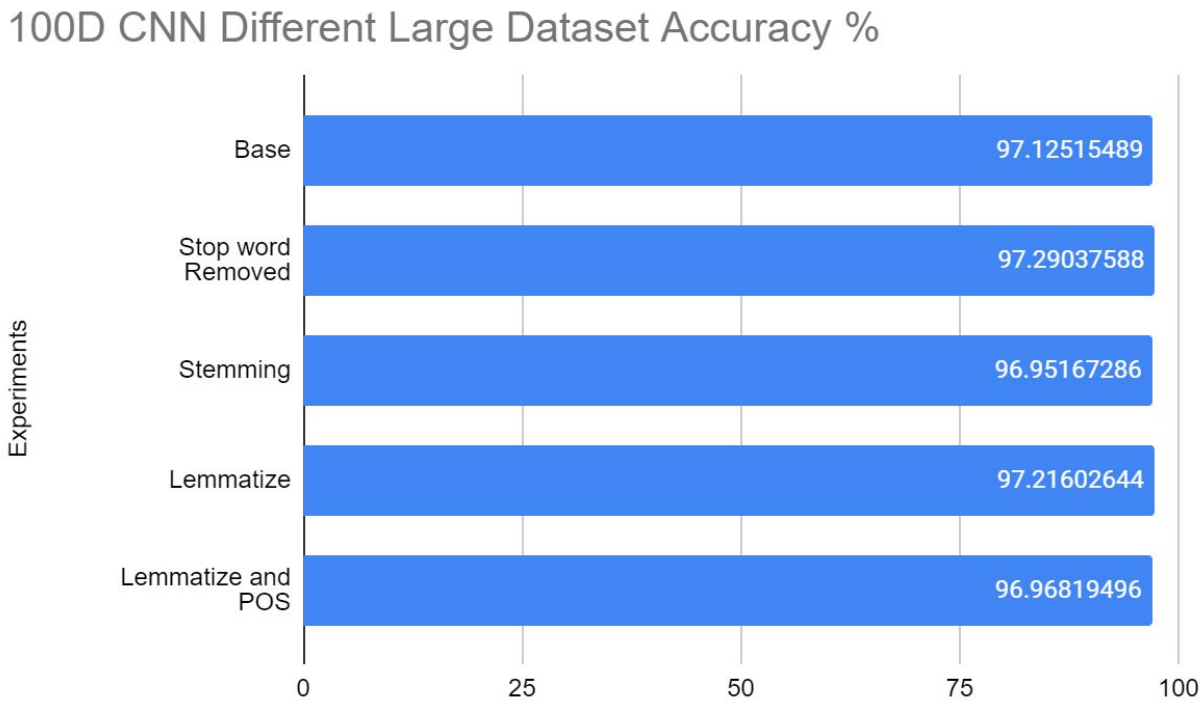

Figure 12: Experiment 8 Data

6.7 Base Model Comparison for all Experiments

The following diagram [Figure 13] shows a comparison of all of the experiments with the base model as the baseline comparison. The logic for this comparison was to show the major differences between the experiments. As noted in the experiment 8 section, the accuracy difference of the larger dataset reveals the importance of data in defining a learning model of natural language processing. The small dataset experiment was the closest to the large dataset

experiment, but that experiment had issues with underfitting. The usage of different learning models and word vectors was a non-factor because they all had similar results.

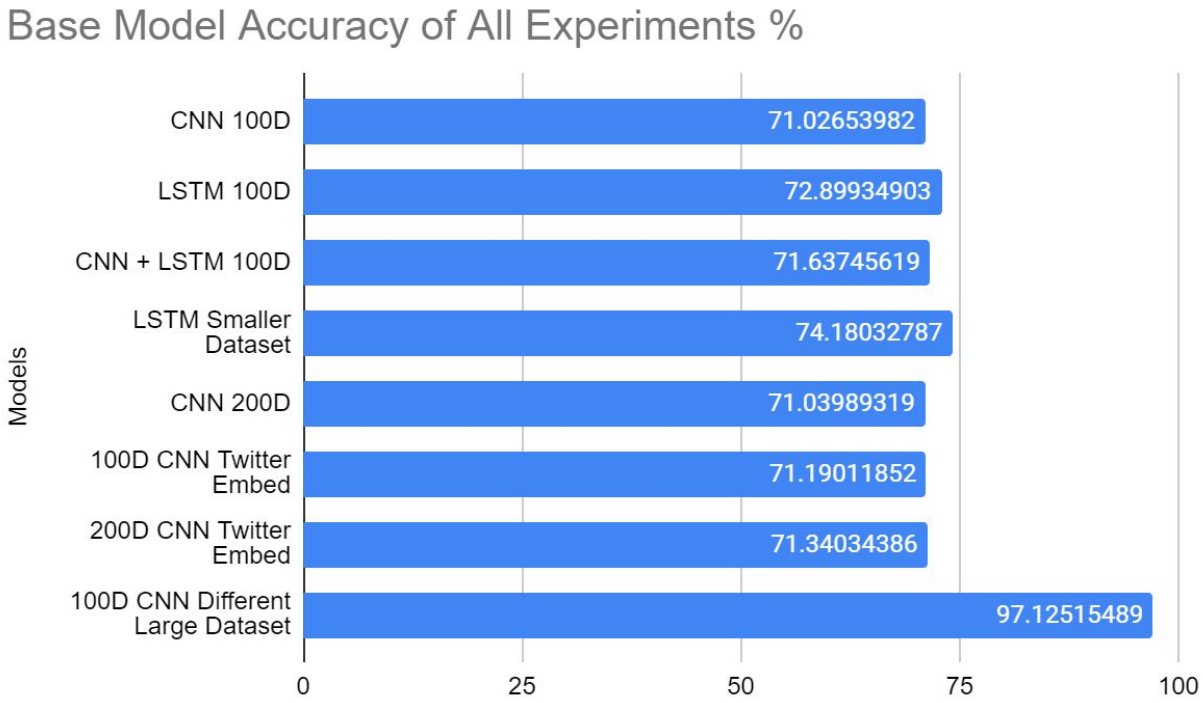

Figure 13:Base model accuracy comparison

6.8 Sentiment Analysis Comparison

The sentiment analysis tests were done on all of the different learning models. The purpose of this test was to verify whether or not sentiment analysis had an impact on the accuracy of the learning model. The following diagram [Figure 14] shows the average accuracy percentage, which reveals that there was an impact on improving the accuracy of the previous linguistic analysis test. The main reason for this difference was probably due to the fact that the learning model had to be slightly modified to compensate for having a multi-label Y data. The Y

data for model construction had an extra column where it included converted sentiment values. The value of 1 represented positive sentiment while 0 represented neutral and negative sentiment values. To keep the rest of the project consistent, the learning model modification was only applied to the sentiment analysis test and multidata test, therefore those two tests will be compared.

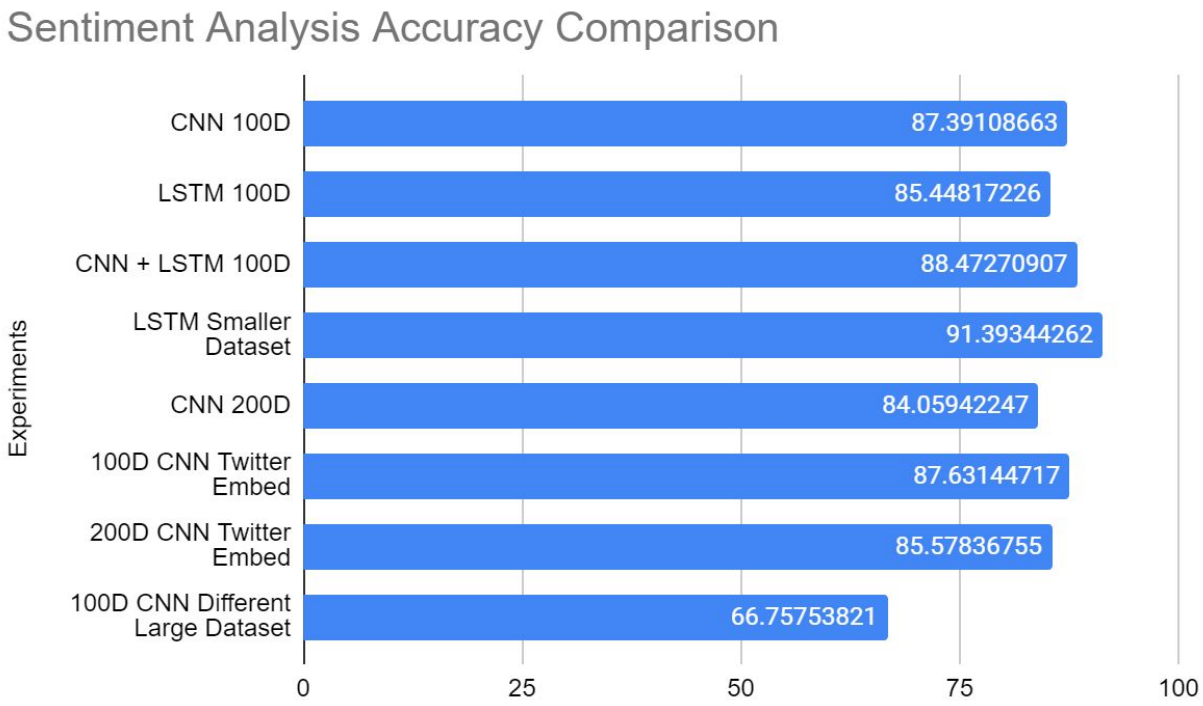

Figure 14:Sentiment analysis accuracy comparison

6.9 Multidata and Combined Model Comparison

The multidata tests followed the sentiment analysis test due to the results that were shown. The logic behind this was to verify if splitting the data into multiple labels would impact the accuracy. The code changes were only applied to the preprocessing section, while the

modeling section had the same learning model as the sentiment analysis. The tables [Appendix I-IX] affirms the idea that there was a significant accuracy change, but in the opposite direction. The drop in accuracy acknowledges the idea that having to deal with multiple labels will negatively affect the data. The cause of this data discrepancy could be due to the fact that a sequential Keras API was used for the modeling section. The sequential API in Keras has some limitations when dealing with multiple outputs, thus to improve on this test, a functional API was used to help mitigate this issue. The multidata tests helped reveal that the combined test needed to use a single binary input and single binary output.

The combined tests utilized a few of the linguistic analysis and combined them together into one python file. The linguistic analysis that was first applied on the data was stopword removal followed by the lemmatize with part of speech. The tables [Appendix I-IX] report the result, where the accuracy percentage was consistently similar for all of the combined tests within their experiment. This would mean that applying linguistic analysis will have a negative to no impact on the learning model. The combined test was a measure on how well linguistic analysis will perform when mixed with other analyses. The logic behind this test was because in most scenarios more than one linguistic analysis will be applied on a dataset. The result reveals, the accuracy percentages did not have a significant drop or rise, thus further confirming that linguistic analysis was not a major factor in the learning model.

6.10 Convergence of Accuracy and Loss Value

The following diagrams [Figure 15, 16] reveals the convergence of the training and testing accuracy rate as it goes through the 12 epochs of the learning model, double the tested

amount. The model and test used for the diagrams was the LSTM base model and combined model. As reported in the diagrams, the accuracy for both of the learning models converges at a constant of around 71%. This fact demonstrates that linguistic analysis had little impact on the learning model because the testing and training accuracy did not increase as more epochs were added to the model. The other purpose of these convergence diagrams was to affirm that the linguistic analysis tests were not overfitting or underfitting. These events may occur when the accuracy rate of the testing set does not converge with the training set. The diagrams [Figure 15, 16] confirms that the rates did converge, which means that the modeling aspect of the experiment was sufficient. Since the combined tests had similar accuracy results as the other single linguistic analysis tests, it can be assumed that other tests will have similar convergence rate.

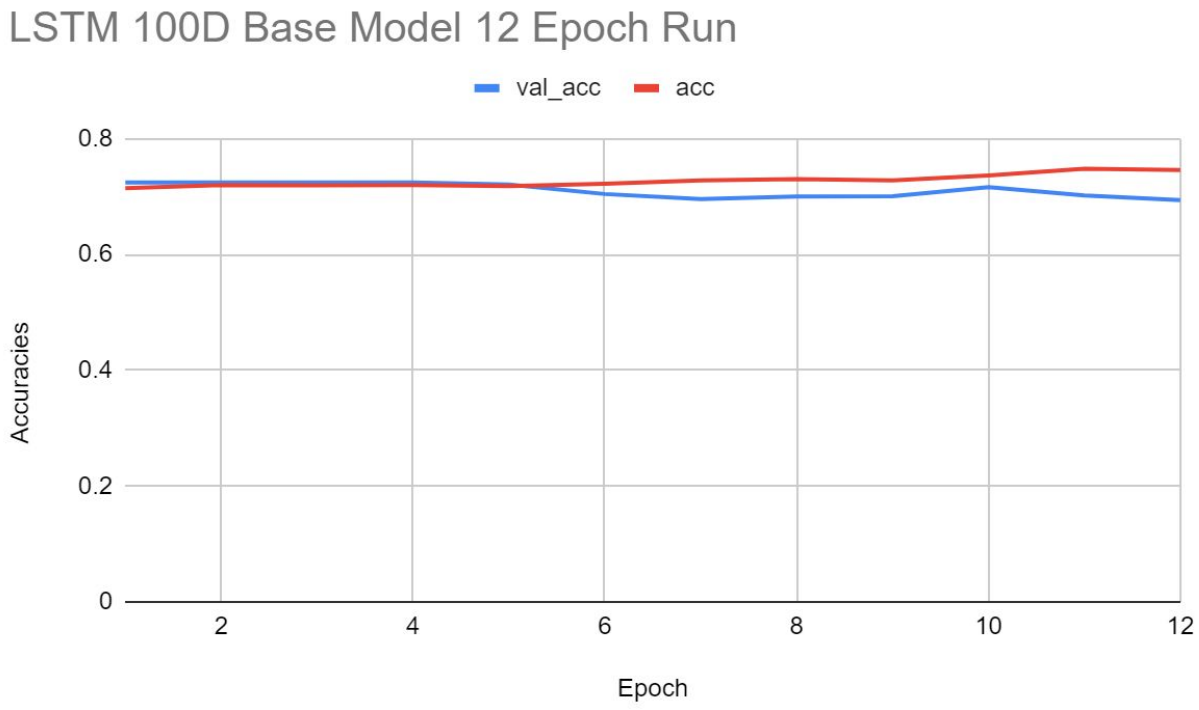

Figure 15: Convergence test on base model

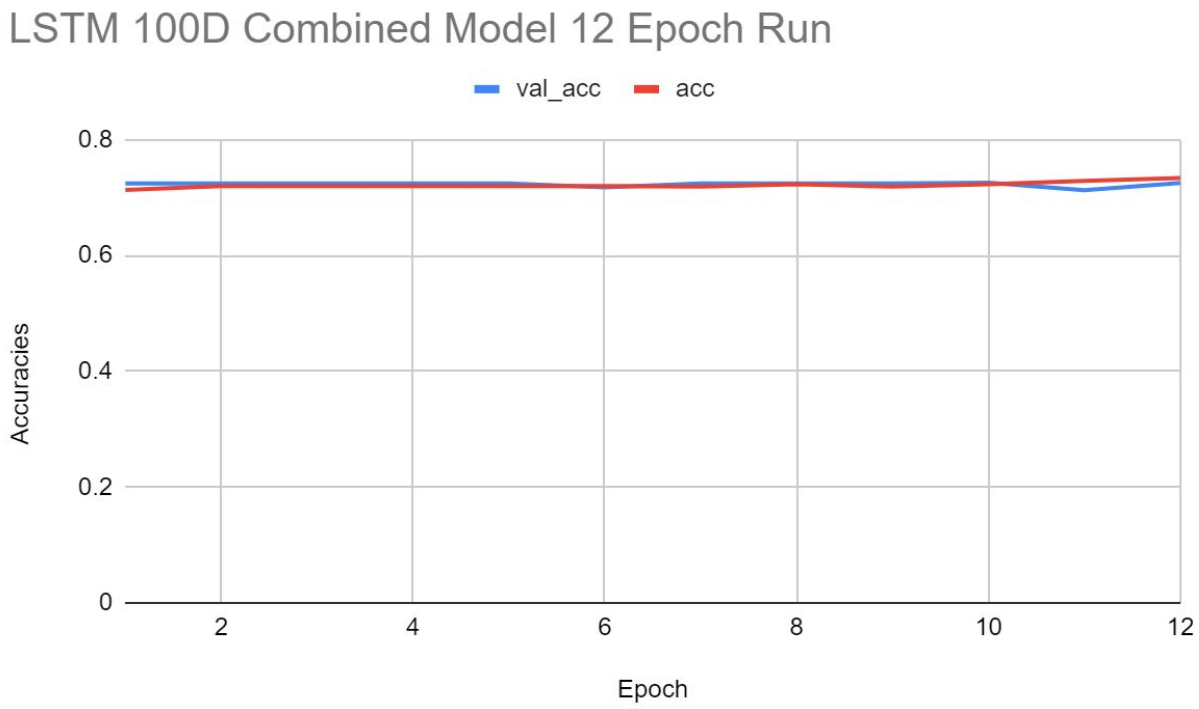

Figure 16: Convergence test on combined model

6.11 Runtime Tests

The following graph [Figure 17, 18] reveals that there was a redeeming factor for applying linguistic analysis to a dataset. For these tests, it was accomplished by measuring the time it took for the learning model to learn the data, while any other preprocessing was taken out of the measurement. The logic behind this was due to the fact that linguistic analysis took longer for larger datasets. The file size for the medium dataset was around 1.5 megabytes and the large dataset was around 53.1 megabytes. The first graph [Figure 17] was tested with the CNN 100D base and combined learning model, and medium sized dataset. The data shows that there was a slight decrease in completion time for when linguistic analysis was applied to the data. The

percentage difference in runtime between the base and combined model was around 3.54%, with the combined model being faster. The second graph [Figure 18] was achieved with the same learning models, but with the larger dataset. This second graph affirmed the idea that linguistic analysis could reduce runtime, since the data had similar results as the first graph. The difference between the base and combined model runtime was around 4.15%, which was a slightly higher percentage than the medium sized dataset results. These results give off a decent indication that if larger datasets were used for modeling, then linguistic analysis would be a method to reduce the runtime. Another comparison to make was the difference in runtime between the large and medium dataset models [Appendix X]. The time it took to complete the modeling process for the large dataset was greater than the medium dataset, therefore for larger datasets a runtime reduction would help reduce a large amount of time for modeling.

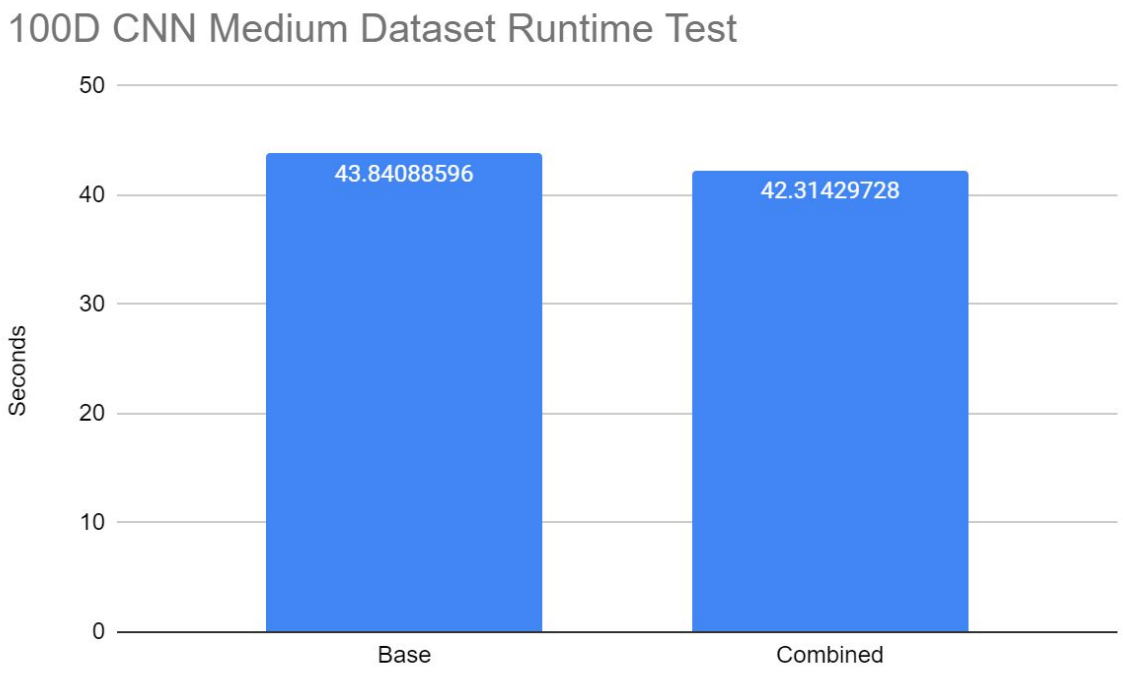

Figure 17: Runtime Test for Medium Dataset

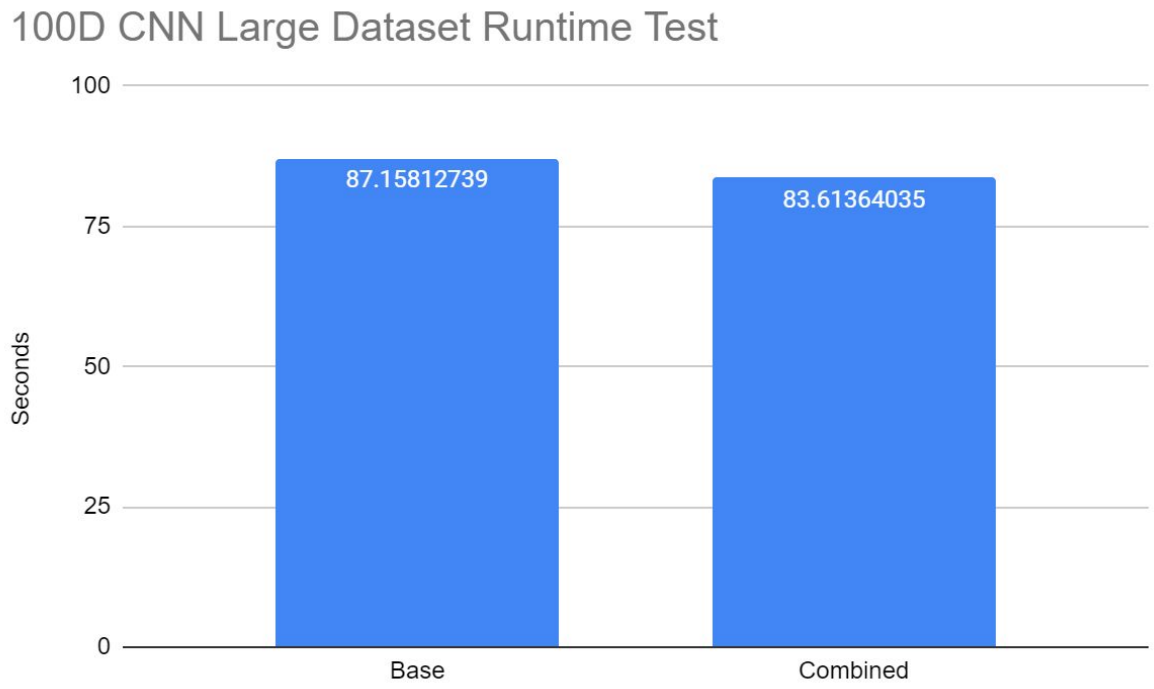

Figure 18: Runtime Test for Large Dataset

6.12 Real World Tests

The last tests applied to the experiment was a real world test that utilized 10 labeled statements from POLITIFACT.com and 5 labeled statements from SNOPES.com. The statements label include false, true, half-true, mostly-true, and pants-fire. The label for barely-true was not included in the real world test set. FakerFact.org was the fake news checker tool that was used to compare with the created neural network models. The FackerFact AI tool was named Walt and had been trained on millions of documents such as scientific journals, satire articles, narrative fiction, and opinion pieces [34]. The AI tool allows users to input statements that will then result in a label of journalism, wiki, satire, sensational, opinion, or agenda-driven. Since the resulting label does not exactly match the labels for the project's learning models, a

slight modification was necessary on the FakerFact labels. The label for journalism and wiki were considered as true news, while satire, sensational, opinion, and agenda-driven were recognized as fake news. The same distinction was applied to the real world test dataset where any statements that were recognized as true were therefore true, while the same was applied for fake statements. The learning model that was used for this test was the LSTM base model, LSTM stopword removed, LSTM lemmatization with part of speech, and CNN large dataset combination model. The latest models run were saved during each test; therefore these models were able to be loaded and used for prediction. A predict() method was applied to the test so that the results given would be the amount that the model predicted correctly. The following diagram [Figure 19] shows the percentage of correct scores when compared to the scores of the FakerFact test. The FakerFact score was at 66% whereas the learning models had scores around 40%. The scores reveal that the learning models were neither consistent nor accurate in trying to determine the fakeness of unseen input statements. A possible scenario that might explain the results was that the real world text statements fell under the group of learning data that failed during the modeling process. One noteworthy aspect of this experiment was during the last two tests where both of them were run without any linguistic analysis applied, which resulted in the same or higher accuracy. The real world test was designed to be a gauge to verify the learning models, but it proved that they were not as consistent as displayed during the modeling process. The main factor to acknowledge was that the modeling process utilized thousands of data tuples to get an accurate percentage, whereas those percentages will vary depending on the data used for the input.

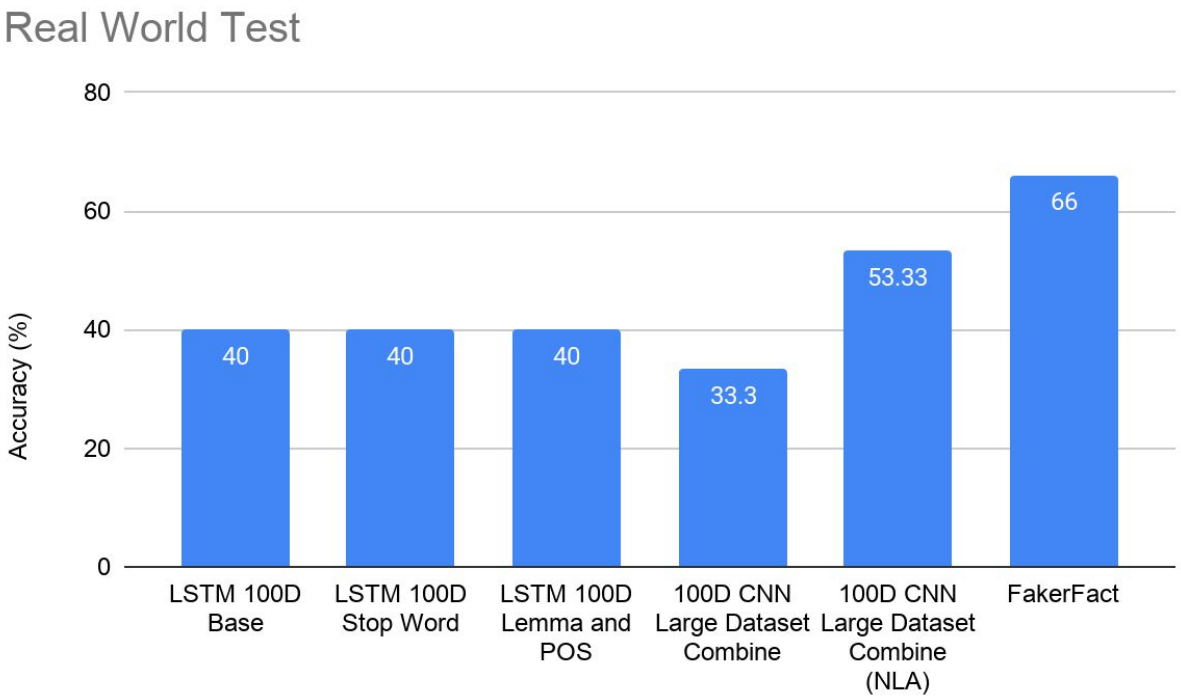

Figure 19: Real World Test Data

Chapter 7

Conclusion and Future Work

In this experiment, it was hypothesized that usage of linguistic analysis will improve the accuracies of learning models for fake news detection. As explained in the data results and analysis section, the hypothesis was proven to be incorrect when applying single and multiple linguistic analysis to a learning model with fake news data. The initial base model accuracy without linguistic analysis was used as a baseline to gauge the other tests that did apply linguistic analysis. Those linguistic tests resulted in marginal gains and loss that was a non-factor when determining the validity of linguistic analysis. The use of different neural network models and word vectors revealed that changing them will also not improve the accuracies of linguistic tests. The only noticeable tests that resulted in a significant impact on the learning model was during the sentiment analysis tests and the large dataset experiment. The accuracies from the large dataset models were around 26% higher, while the linguistic analysis tests, relative to their own experiment, averaged around similar accuracies. This result further ingrained the concept that linguistic analysis was a non-factor in training a neural network model for fake news. To verify this new concept, a real world scenario test was designed to compare a third party fake news detector tool and various learning models created during the experiments. The results from the real world test confirmed that applying linguistic analysis to learning models did not improve the classification accuracy when compared to the base model and third party detector. Even though linguistic analysis could not be used to improve the dataset for a deep learning model, the result from this project shows that large datasets are more valuable in model construction, therefore it is best to focus on gathering data rather than dataset improvements. A way that linguistic

analysis could be used in this area is to reduce runtime for modeling larger datasets. The runtime tests [Figure 17, 18] showed that there was a reduction in runtime when linguistic analysis was applied before the modeling process.

Due to the limitation of natural language processing, neural networks will not be able to singlehandedly solve the fake news issue, but if an application was built with modeling as a backbone, then consistent prediction could be achieved on any sized data. An application in mind was to use the learning models in this experiment as preprocessing tools for larger modeling applications. An example would be to use the project's learning models as early detectors in model construction where the models would be used to reclassify data. If the inputted data were incorrectly classified then the data would be further processed and learned in a new model. That newly built model and this project's model will be thus used in tandem as a possible solution to help consistently predict unseen data. In conclusion, linguistic analysis may not be the best method to improve modeling, but there are ways where linguistic analysis could be used for data collection, preprocessing, and performance improvements.

References

[1] PsPrint.com. The invention and history of the printing press. https://www.psprint.com/resources/printing-press/, 2019. [Online; accessed 26-November-2019].

[2] Wikipedia contributors. Fake news — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Fake_news, 2019. [Online; accessed 27-November-2019].

[3] Claire Wardle. Fake news. it's complicated. https://firstdraftnews.org/latest/fake-news-complicated/, 2017. [Online; accessed 27-November-2019].

[4] Jeffrey M. Jones. U.s. media trust continues to recover from 2016 low. https://news.gallup.com/poll/243665/media-trust-continues-recover-2016-low.aspx, 2018. [Online; accessed 27-November-2019].

[5] Samantha Putterman. No evidence that cream of tartar and orange juice will help you quit smoking. https://www.politifact.com/facebook-fact-checks/statements/2019/oct/30/facebook-posts/no-evidence-cream-tartar-orange-juice-drink-will-h/, 2019. [Online; accessed 27-November-2019].

[6] Alayna Degenhardt, Farrah Frattaroli. How many west virginia students are homeless? https://www.politifact.com/west-virginia/statements/2019/oct/11/joe-manchin/how-many-west-virginia-students-are-homeless/, 2019. [Online; accessed 27-November-2019].

[7] GeeksforGeeks.com. Turing test in artificial intelligence. https://www.geeksforgeeks.org/turing-test-artificial-intelligence/, 2017. [Online; accessed 28-November-2019].

[8] Igor Bobriakov. Artificial intelligence vs. machine learning vs. deep learning. what is the difference? https://medium.com/activewizards-machine-learning-company/artificial-intelligence-vs-machine-learning-vs-deep-learning-what-is-the-difference-a5e2bc8b835f, 2019. [Online; accessed 28-November-2019].

[9] Margaret Rouse. machine learning (ml). https://searchenterpriseai.techtarget.com/definition/machine-learning-ML, 2018. [Online; accessed 29-November-2019].

[10] Margaret Rouse. Ai (artificial intelligence). https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence, 2018. [Online; accessed 29-November-2019].

[11] Margaret Rouse. deep learning. https://searchenterpriseai.techtarget.com/definition/deep-learning-deep-neural-network, 2018. [Online; accessed 29-November-2019].

[12] Kendall Fortney. Pre-processing in natural language machine learning. https://towardsdatascience.com/pre-processing-in-natural-language-machine-learning-898a84b8bd47, 2017. [Online; accessed 29-November-2019].

[13] S. Girgis, E. Amer, and M. Gadallah. Deep learning algorithms for detecting fake news in online text. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pages 93–97, Dec 2018.

[14] thiagorainmaker77. liar_dataset. https://github.com/thiagorainmaker77/liar_dataset/, 2019. [Online; accessed 29-November-2019].

[15] Hicham EL BOUKKOURI. Arithmetic properties of word embeddings. https://medium.com/data-from-the-trenches/arithmetic-properties-of-word-embeddings-e918e3fda2ac, 2018. [Online; accessed 29-November-2019].

[16] A. Dey, R. Z. Rafi, S. Hasan Parash, S. K. Arko, and A. Chakrabarty. Fake news pattern recognition using linguistic analysis. In *2018*

Joint 7th International Conference on Informatics, Electronics Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision Pattern Recognition (icIVPR), pages 305–309, June 2018.

[17] Megan Risdal. Getting real about fake news. https://www.kaggle.com/mrisdal/fake-news, 2017. [Online; accessed 23-March-2020].

[18] Keras.io. Keras: The python deep learning library. https://keras.io/, 2020. [Online; accessed 23-March-2020].

[19] Dhruvil Karani. Introduction to word embedding and word2vec. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa, 2018. [Online; accessed 23-March-2020].

[20] Jason Brownlee. What are word embeddings for text? https://machinelearningmastery.com/what-are-word-embeddings/, 2017. [Online; accessed 23-March-2020].

[21] Prabhu. Understanding of convolutional neural network (cnn) deep learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, 2018. [Online; accessed 23-March-2020].

[22] Uniqtech. Multilayer perceptron (mlp) vs convolutional neural network in deep learning. https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1, 2018. [Online; accessed 23-March-2020].

[23] Denny Britz. Understanding convolutional neural networks for nlp. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/, 2015. [Online; accessed 23-March-2020].

[24] Aditi Mittal. Understanding rnn and lstm. https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e, 2019. [Online; accessed 23-March-2020].

[25] Michael Nguyen. Illustrated guide to lstms and grus: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018. [Online; accessed 23-March-2020].

[26] Jason Brownlee. Cnn long short-term memory networks. https://machinelearningmastery.com/cnn-long-short-term-memory-networks/, 2017. [Online; accessed 23-March-2020].

[27] Lima Vallantin. Why is removing stop words not always a good idea. https://medium.com/@limavallantin/why-is-removing-stop-words-not-always-a-good-idea-c8d35bd77214, 2019. [Online; accessed 23-March-2020].

[28] Saurav Jain. Introduction to stemming. https://www.geeksforgeeks.org/introduction-to-stemming/, 2018. [Online; accessed 23-March-2020].

[29] Tushar Srivastava. Nlp: A quick guide to stemming. https://medium.com/@tusharsri/nlp-a-quick-guide-to-stemming-60f1ca5db49e, 2019. [Online; accessed 23-March-2020].

[30] Yash_R. Python lemmatization with nltk. https://www.geeksforgeeks.org/python-lemmatization-with-nltk, 2018. [Online; accessed 23-March-2020].

[31] Hunter Heidenreich. Stemming? lemmatization? what? https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8, 2018. [Online; accessed 23-March-2020].

[32] MonkeyLearn. Sentiment analysis. https://monkeylearn.com/sentiment-analysis/, 2020. [Online; accessed 23-March-2020].

[33] Christopher D. Manning Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation. https://nlp.stanford.edu/projects/glove/, 2015. [Online; accessed 23-March-2020].

[34] FakerFact.com. About fakerfact. https://www.fakerfact.org/about, 2017. [Online; accessed 23-March-2020].

Appendix I

Experiment 1: CNN 100D								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.7090635955	0.7265898849	0.7050575865	0.719078618	0.7155733602	0.8387581373	0.2208312469	0.7220831248
Run 2	0.7165748624	0.7160741113	0.7285928894	0.719078618	0.7215823737	0.8838257387	0.2188282424	0.6995493241
Run 3	0.695042564	0.7095643467	0.7185778669	0.7240861293	0.7305958939	0.8818227341	0.2428642965	0.6935403106
Run 4	0.7205808714	0.7270906361	0.7250876315	0.7105658489	0.6815222835	0.8713069605	0.2373560342	0.6544817227
Run 5	0.7100650978	0.7160741113	0.7145718579	0.7160741113	0.7120681023	0.8938407612	0.2218327492	0.7135703556
Average Accuracy (*100)	71.02653982	71.90786181	71.83775665	71.77766651	71.22684027	87.39108663	22.83425138	69.66449676
Standard Deviation	0.009739577196	0.007569462407	0.009237467194	0.004952122301	0.01856699826	0.02121672536	0.01097090752	0.02612174382

Appendix II

Experiment 2: LSTM 100D								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.7315973962	0.7285928894	0.7315973962	0.7310966451	0.7315973962	0.8678017026	0.2243365048	0.7285928894
Run 2	0.7315973962	0.7250876316	0.7225838759	0.7315973962	0.7315973962	0.845768653	0.2318477717	0.7315973962
Run 3	0.7315973962	0.7315973962	0.7310966451	0.7315973962	0.7305958939	0.8622934402	0.2178267401	0.7320981473
Run 4	0.7210816226	0.7280921383	0.7315973962	0.7320981473	0.7315973962	0.8798197296	0.2203304958	0.7310966451
Run 5	0.7290936406	0.7315973962	0.7315973962	0.7290936406	0.7200801203	0.8167250877	0.2073109665	0.7325988984
Average Accuracy (*100)	72.89934903	72.89934903	72.96945419	73.10966451	72.90936406	85.44817226	22.03304958	73.11967953
Standard Deviation	0.004553807716	0.002728978755	0.004429595882	0.001174365488	0.005057338477	0.02440256368	0.009006562721	0.001559580521

Appendix III

Experiment 3: CNN + LSTM 100D								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.6820230346	0.7045568354	0.6885327993	0.7120681023	0.7200801203	0.8577866801	0.2373560341	0.730595894
Run 2	0.7315973962	0.703555333	0.6905358038	0.7280921383	0.727090636	0.9023535303	0.2313470206	0.7170756135
Run 3	0.7341011518	0.6955433152	0.7200801203	0.7120681023	0.7275913872	0.8903355033	0.2218327491	0.7025538308
Run 4	0.7095643466	0.7230846271	0.6805207814	0.7336004007	0.7195793692	0.9068602904	0.2348522785	0.7260891338
Run 5	0.7245868804	0.7165748624	0.7175763646	0.7010515775	0.7180771158	0.8662994492	0.2353530296	0.727090636
Average Accuracy (*100)	71.63745619	70.86629946	69.94491739	71.73760642	72.24837257	88.47270907	23.21482224	72.06810216
Standard Deviation	0.02144653414	0.01101880068	0.01810476758	0.01323726865	0.004498406534	0.02177950809	0.006159442414	0.01136293205

Appendix IV

Experiment 4: LSTM Smaller Dataset								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.9139344262	0.1926229509	0.7418032787
Run 2	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.9139344262	0.1926229509	0.7418032787
Run 3	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.9139344262	0.1926229509	0.7418032787
Run 4	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.9139344262	0.1926229509	0.7418032787
Run 5	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.7418032787	0.9139344262	0.1926229509	0.7418032787
Average Accuracy (*100)	74.18032787	74.18032787	74.18032787	74.18032787	74.18032787	91.39344262	19.26229509	74.18032787
Standard Deviation	0	0	0	0	0	0	0	0

Appendix V

Experiment 5: CNN 200D								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.7020530797	0.7075613422	0.7005508264	0.7300951428	0.7240861293	0.8602904357	0.2203304958	0.7175763647
Run 2	0.7110666001	0.7260891338	0.7105658489	0.7255883827	0.7125688534	0.8132198298	0.2188282424	0.7005508264
Run 3	0.7180771158	0.7165748624	0.679519279	0.6860290437	0.703054582	0.8482724087	0.2178267401	0.7015523287
Average Accuracy (*100)	71.03989319	71.67417795	69.68786514	71.39041897	71.32365216	84.05942247	21.89951594	70.65598399
Standard Deviation	0.008032855563	0.009265023594	0.01584569426	0.02424552632	0.01053165851	0.024456622214	0.001260196055	0.009553722582

Appendix VI

Experiment 6: 100D CNN Twitter Embed								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.7285928895	0.7330996496	0.7130696045	0.7336004007	0.7015523286	0.8913370055	0.2333500251	0.7245868804
Run 2	0.7115673511	0.7085628444	0.7270906361	0.7341011518	0.7105658489	0.8903355033	0.2328492739	0.727090636
Run 3	0.6955433151	0.7240861293	0.7200801203	0.7170756135	0.6810215324	0.8472709064	0.2248372559	0.7240861293
Average Accuracy (*100)	71.19011852	72.19162077	72.00801203	72.82590553	69.77132367	87.63144717	23.03455183	72.52545486
Standard Deviation	0.01652731602	0.01241149152	0.007010515789	0.009688380478	0.01514168517	0.02515744953	0.004776861316	0.00160968965

Appendix VII

Experiment 7: 200D CNN Twitter Embed								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.7250876315	0.7110666	0.7160741113	0.6970455684	0.7220831248	0.8172258388	0.2303455183	0.7170756135
Run 2	0.7255883827	0.7115673511	0.7040560842	0.7255883827	0.727090636	0.8622934402	0.2298447672	0.7280921383
Run 3	0.6895343015	0.7085628444	0.7305958939	0.7090635955	0.7110666	0.8878317477	0.2178267401	0.6730095144
Average Accuracy (*100)	71.34034386	71.03989319	71.69086965	71.05658489	72.00801203	85.57836755	22.60056752	70.60590888
Standard Deviation	0.02067279529	0.001609689668	0.01328957389	0.0143305839	0.008197649213	0.03575026389	0.007087589284	0.02914698849

Appendix VIII

Experiment 8: 100D CNN Different Large Dataset								
	Base	**Stop word Removed**	**Stemming**	**Lemmatize**	**Lemmatize and POS**	**Sentiment**	**Multidata**	**Combine**
Run 1	0.9727385378	0.9700123916	0.9685254027	0.9737298637	0.9700123916	0.6718711276	0.7254027261	0.9677819083
Run 2	0.9727385378	0.9757125155	0.9695167286	0.9710037175	0.9697645601	0.6671623296	0.726394052	0.9717472119
Run 3	0.9682775713	0.9729863693	0.9705080545	0.9717472119	0.9692688971	0.663692689	0.7204460967	0.9719950434
Average Accuracy (*100)	97.12515489	97.29037588	96.95167286	97.21602644	96.96819496	66.75753821	72.40809583	97.05080545
Standard Deviation	0.002575540234	0.002850959757	0.000991325898 4	0.001409229418	0.000378568830 6	0.004104835 44	0.003186666 918	0.002364161 59

Appendix IX

Real World Test	* No linguistic analysis on prediction file					
	LSTM 100D Base	LSTM 100D Stop Word	LSTM 100D Lemma and POS	100D CNN Large Dataset Combine	100D CNN Large Dataset Combine (NLA)	FakerFact
Accuracy (%)	40	40	40	33.3	53.33	66

Appendix X

Runtime Test				
	100D CNN Medium Dataset Base (seconds)	**100D CNN Medium Dataset Combined (seconds)**	**100D CNN Large Dataset Base (seconds)**	**100D CNN Large Dataset Combined (seconds)**
Run 1	43.55672693	42.17908287	86.55423498	82.23058093
Run 2	43.18384695	42.13161492	86.34090805	84.14311302
Run 3	44.78208399	42.63219404	88.57923913	84.46722711
Average	43.84088596	42.31429728	87.15812739	83.61364035
Percentage Difference	3.54%	3.54%	4.15%	4.15%